**COMPILE CONSTRUCTION LAB FINAL**

**SUBMITTED BY: DUA NADEEM**

**SUBMITTED TO: SIR. BILAL HAIDER**

**REGISTRATION N0: FA21-BCS-016**

**Question#3**

**Give examples of four optimizations in your mini compiler.**

Here are four examples of optimizations that are implemented in the mini compiler:

**1. Constant Folding**

**Description:** Pre-compute expressions involving constants during compilation rather than at runtime.  
**Example Before Optimization:**

plaintext

Copy code

x = 5 + 10;

y = x \* 2;

**Optimized Output:**

plaintext

Copy code

x = 15;

y = 30;

**Benefit:** Reduces runtime computation, making the program more efficient.

**2. Dead Code Elimination**

**Description:** Remove code that does not affect the program's output or functionality.  
**Example Before Optimization:**

plaintext

Copy code

x = 10;

y = 20;

z = x + y; // z is calculated but never used

**Optimized Output:**

plaintext

Copy code

x = 10;

y = 20;

**Benefit:** Minimizes resource usage and makes the program more concise.

**3. Loop Invariant Code Motion**

**Description:** Move calculations that do not change within a loop to outside the loop.  
**Example Before Optimization:**

plaintext

Copy code

for (i = 0; i < 10; i++) {

x = 5 \* 2; // Invariant calculation

y = i + x;

}

**Optimized Output:**

plaintext

Copy code

x = 10; // Moved outside the loop

for (i = 0; i < 10; i++) {

y = i + x;

}

**Benefit:** Reduces repeated computations inside the loop, improving performance.

**4. Strength Reduction**

**Description:** Replace expensive operations with equivalent but cheaper ones.  
**Example Before Optimization:**

plaintext

Copy code

y = x \* 8; // Multiplication by a power of 2

**Optimized Output:**

plaintext

Copy code

y = x << 3; // Bitwise left shift (faster operation)

**Benefit:** Reduces computational cost, especially in resource-constrained environments.